

# A First Look at Data Center Network Conditions Through The Eyes of PTPmesh

Diana Andreea Popescu  
University of Cambridge  
diana.popescu@cl.cam.ac.uk

Andrew W. Moore  
University of Cambridge  
andrew.moore@cl.cam.ac.uk

**Abstract**—Increased network latency and packets losses can affect substantially application performance. Due to the scale of data centers, custom network monitoring tools have been developed to measure network latency and packet loss. In our previous work, we used the Precision Time Protocol (PTP) to measure one-way delay and to quantify packet loss ratios, and we proposed PTPmesh as a cloud network monitoring tool. In this work, we provide a better understanding on how to exploit the measurement data offered by PTPmesh and present a detailed analysis of PTPmesh measurements collected in ten data centers from three cloud providers. Our findings reveal different latency, latency variance and packet loss characteristics across data centers. Through our analysis, we showcase the strengths and limitations of PTPmesh as a cloud network monitoring tool. To foster further research in this area, we make our dataset available.

**Index Terms**—PTP, Network Measurement, Data Centers, Network Latency, One-way Delay, Packet Loss

## I. INTRODUCTION

With more and more businesses moving to the cloud, the data center networks [1], [2] that power up the cloud have to be reliable and to offer performance guarantees. Data center networks have become increasingly complex in recent years, requiring a significant effort to debug and troubleshoot. Furthermore, application performance can be affected by network conditions in data centers [3]–[5]. Several data center network monitoring tools have been proposed, mostly by cloud operators [6]–[8]. However, for their customers, it is usually difficult to have access to information regarding network conditions. In our previous work, we introduced *PTPmesh* [9] as an easy to deploy network monitoring tool for cloud users to gather information such as network latency and packet losses. PTPmesh is a lightweight measurement service that is always-on and that offers end-to-end one-way delay (OWD) measurements. In this paper, we calibrate the open source software implementation of PTP, the PTPd [10] daemon, by performing several experiments both on local testbeds and in the cloud. We use PTPmesh to perform network latency and packet loss measurements in data centers from different cloud providers (Amazon AWS EC2, Google Compute Engine, Microsoft Azure), in order to validate its use as a monitoring tool for measuring network conditions with network traffic from data centers. We showcase the strengths and limitations of PTPmesh through analysis of the measurement data.

In this paper, we make the following contributions:

- We analyze the impact of virtualization on the OWD measurement, the overhead of PTPd at the end-host and on the network depending on the message frequency, and the impact the number of PTPd clients has on the accuracy of OWD.
- We perform measurements between pairs of virtual machines (VMs) in ten different data centers from three cloud providers, and we perform a temporal analysis of the traces at different time scales. We identify different characteristics with regards to latency, latency variance and packet loss for data centers.
- We share publicly our datasets at <https://doi.org/10.17863/CAM.23126>.

## II. BACKGROUND AND RELATED WORK

In this section, we summarise our previous work from [9], and we review the related work that has been published since.

**Background.** In our previous work [9], we leveraged the Precision Time Protocol (PTP) [11] to infer network latency and packet loss in data centers through controlled experiments on small-scale testbeds. We used the open source software implementation of PTP, the PTPd daemon. Additionally, we conducted a small-scale study of network conditions in several data centers, albeit limited in temporal and spatial coverage, and presented a preliminary analysis of the collected data. In this work, we collected extensive data and performed a thorough analysis on several dimensions.

For *network latency*, we validated the use of the *one-way delay* as a measure for network latency through experiments that compare the one-way delay with the output of *ping* and a custom UDP-based tool [4]. We also studied the effect of network congestion on the one-way delay, showing that through increases in the one-way delay values, we can detect periods of network congestion. The duration of the network congestion periods that can be detected depends on the message frequency. The one-way delay also has increased values after the network congestion period has passed because of the time needed for the slave’s clock to converge again. We also showed that using PTP-enabled NICs (Solarflare SFN8552 Network Interface Card (NIC) [12] supporting PTP with hardware timestamping NICs and using the *sfptpd* daemon) removes the overhead and inaccuracies introduced by a software solution, since the packets are timestamped at the NIC, leaving the one-way delay measurement to account only for the delay within

the network. For *packet losses*, we proposed and validated a metric for computing packet loss ratio over a period of time. We defined the metric based on the number of *Delay Request* and *Delay Response* messages, and we computed it as

$$1 - \frac{\#Delay\_Response\_messages}{\#Delay\_Request\_messages}.$$

**Related work.** Network monitoring for data centers is an active area of research, and several works have appeared since our previous publication [8], [13]–[16]. deTector [8] presents an algorithm to minimize the number of probes sent for detecting and localizing packet losses and latency spikes. [13], [14] and [15] employ passive measurement for network faults localization. [13] presents a classification algorithm that identifies the root cause of failure using TCP statistics collected at one of the endpoints. The work in [14] looks from the end-host to identify the faulty links and switches, by correlating anomalies in end-host statistics with the network path of the traffic. Vigil [15] tracks the path of TCP connections that display retransmissions through traceroute, and identifies the links with the most retransmissions as the faulty ones. [16] presents a sampling framework which can poll a subset of switch counters at microsecond-level granularity, used to determine microbursts in data centers. Our work can determine periods of increased latency with millisecond-level granularity depending on the message frequency. Increasing the message frequency beyond 128 messages per second allows detection at an even higher resolution. NTP statistics have been used to measure Internet latency [17].

### III. METHODOLOGY AND MEASUREMENT CALIBRATION

**Virtualization.** The experiments run on the local testbed from our previous paper [9] were performed without virtualization, on bare metal hosts. In order to be able to interpret our collected data in the cloud, where virtualization is the norm, we ran an experiment on the same local testbed to quantify the overhead of virtualization on PTPd statistics, with the PTPd master and client running in VMs. The hypervisor used is Oracle VM VirtualBox version 5.0.40 Ubuntu115130, which is a hosted hypervisor. Without virtualization, the OWD has the following distribution: median 83.14 $\mu$ s, average 83.24 $\mu$ s, standard deviation 1.15 $\mu$ s, 99<sup>th</sup> percentile 85.47 $\mu$ s, 99.9<sup>th</sup> percentile 85.59 $\mu$ s, minimum value 80.85 $\mu$ s, maximum value 85.63 $\mu$ s. When PTPd runs in VMs, median OWD reaches 285.28 $\mu$ s, average 285.37 $\mu$ s, standard deviation 5.44 $\mu$ s, 99<sup>th</sup> percentile 295.64 $\mu$ s, 99.9<sup>th</sup> percentile 295.9 $\mu$ s, minimum value 273.96 $\mu$ s, maximum value 295.92 $\mu$ s. We exclude the first 5 minutes from the data, when the two clocks are not synchronized. These results show that virtualization adds almost 200 $\mu$ s of overhead and increased jitter to the OWD, although the standard deviation of the OWD is not significant. Both these issues can be solved by using PTP-enabled NICs, which provide hardware timestamping, as we showed in our previous work. More recently, OS bypass through a custom software packet processing path [18] or through custom hardware [2] alleviates these issues. Also, given that some cloud providers offer bare metal instances, the virtual switch overhead would be removed in this case.

**Cloud Experimental Setup.** We used PTPd v2 2.3.1, using the latest source code from the public repository. We ran PTPd in unicast mode, using unicast negotiation, and using end-to-end delay measurement. Since currently multicast either requires additional configuration and expenses in the case of Amazon AWS<sup>1</sup> or is not supported at all in the case of Google Compute Engine<sup>2</sup> and Microsoft Azure<sup>3</sup>, we used PTPd in unicast mode for the cloud deployment. We measured the one-way delay between multiple virtual machines (VMs) from different cloud providers. For each of the three cloud providers chosen, Amazon AWS, Google Compute Engine and Microsoft Azure, we chose several zones, and we rented two, four or ten VMs in each zone. We ran the PTPd master on one VM, while the other VMs acted as slaves, running simultaneously. The VMs' types and specifications, running Ubuntu 16.04, can be seen in Table I. We list them along with an assigned name that we use to identify the traces we collected. For Amazon AWS, we ran measurements in regions Ireland zone eu-west-1a (EC2-EUW), Northern California zone us-west-1b (EC2-USW) and Ohio zone us-east-2a (EC2-USE). For Google Compute Engine, the measurements were ran in us-west1-b (GCE-USW) and europe-west1-b (GCE-EUW), and between us-west1-b (GCE-USW) and us-west1-d (GCE-USW2). For Microsoft Azure, we used UK West (Azure-UKW), UK South (Azure-UKS), US West (Azure-USW) and Korea South (Azure-KS). In UK South, the VM type we used was the Standard D2s v3 and Standard\_E16s\_v3 or E32s (with [2] enabled), while in the other zones we used the Standard D1 v2. We will refer to a zone as a data center in the rest of the paper.

**PTPd overhead.** According to the PTP standard, the interval between messages can be set between  $2^{-7}$  to  $2^7$  seconds, which means a maximum of 128 messages per second. PTPd's experimental implementation allows message frequencies of up to  $2^{30}$  messages per second. We performed several experiments where we vary the number of messages between 1 to  $2^7$  per second to determine whether a different message interval yields different latency values. We set these values for both the number of Sync and Delay Request messages exchanged between the master and the slave. This means that the number of Delay Response messages should be the same as the number of Delay Request messages. In Figure 1, as we increase the message frequency per second, the one-way delay decreases, going from median 262.92 $\mu$ s and 99<sup>th</sup> percentile 286.6 $\mu$ s for 1 message per second, to 191.49 $\mu$ s and 99<sup>th</sup> percentile 237.85 $\mu$ s for 128 messages per second. We hypothesize that the cause of this behaviour is that, when the message frequency increases, the code that performs the timestamps remains in the cache, leading to smaller OWD values. Another cause might be due to the way the interrupts are coalesced at the NIC, since messages are not timestamped by the NIC, but in software. The hypervisor could also be a factor that leads to this behaviour.

<sup>1</sup><https://aws.amazon.com/marketplace/pp/B071RMCZ1X>

<sup>2</sup>[https://cloud.google.com/vpc/docs/vpc/#quotas\\_and\\_limits](https://cloud.google.com/vpc/docs/vpc/#quotas_and_limits)

<sup>3</sup><https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-faq>

TABLE I: VM types and settings for the three cloud providers

Cloud provider	Instance type	vCPU	Mem (GB)	Storage (GB)	Storage Type	#NICs used	Network bandwidth
Amazon AWS	t2.micro	1	1	10/30	Elastic Block Store <sup>4</sup>	1	moderate
Google Compute Engine	n1-standard-1	1	3.75	10	Standard Persistent Disk	1	maximum 2Gbps
Microsoft Azure	Standard D1 v2	1	3.50	50	Local SSD	1	moderate
Microsoft Azure	Standard D2s v3	2	8	50	Local SSD	1	moderate
Microsoft Azure	Standard_E16/32s_v3	16/32	128/256	256/512	Local SSD	1	high

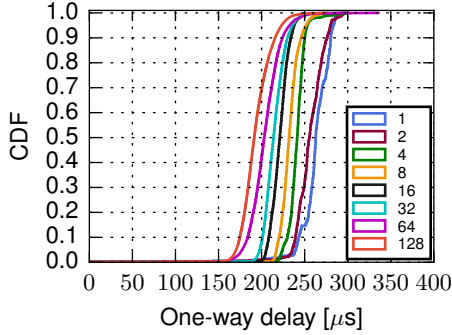


Fig. 1: OWD measured using PTPd for periods of 15 minutes between two VMs.

While increasing the message frequency leads to better accuracy for the one-way delay measurements, the CPU utilisation and network bandwidth consumption increase as well. Since the initial goal was to have a low-overhead measurement system that runs as a service in a VM or in the hypervisor, choosing the message interval implies a tradeoff between host and network resource consumption and measurement accuracy. To this end, we performed an experiment with a PTPd master and a single PTPd client running in the Azure-KS data center with different message frequencies. We ran measurements for each message frequency from 1 message per second to  $2^{30}$  messages per second for 15 minutes. We monitor the average CPU utilisation, memory and send and receive network bandwidth (using *iftop*). The CPU utilisation doubles with the doubling of the message frequency, the maximum CPU utilisation achieved being less than 10%. For a frequency of 1 message per second, the average CPU utilisation is almost 0% and the peak network bandwidth consumption is 576 b/s (receive) and 3.33 Kb/s (send). For a frequency of  $2^7$  messages per second, the average CPU utilisation is 0.7% and the peak network bandwidth consumption is 72.3 Kb/s (receive) and 234 Kb/s (send). From  $2^{12}$  upwards, due to the fact that the protocol operates in a request-response manner and taking into account the latency on the network, the message frequency does not actually achieve the set message frequency. The VM has a network bandwidth of 0.75 Gb/s, which is not reached for message frequencies greater than  $2^{12}$ , the maximum bandwidth consumed being less than 10 Mb/s. Since these values are reported for a single slave, when synchronising with multiple slaves, we expect the network bandwidth to increase proportionally. For example, if using

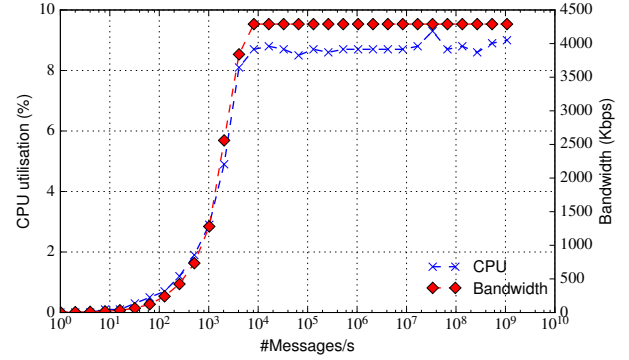


Fig. 2: CPU utilisation and network bandwidth of the PTPd master synchronising with one PTPd client.

1000 slaves, the peak network bandwidth at the PTP master would be 7.23 Mb/s (receive) and 23.4 Mb/s (send). The memory usage is the same regardless of the message interval, being 0.1% when using a VM with 1.6 GB RAM.

**Number of concurrent PTPd clients in the cloud.** Another aspect that needs to be taken into account is the number of slaves a master can synchronize with before becoming overloaded because of CPU processing of messages and end-host queueing of packets. To see if the number of clients affects significantly the OWD values, we performed a suite of experiments in our local testbed with ten bare metal hosts, using one PTPd master and a maximum of nine PTPd clients, and a similar experiment in EC2-USE, using 128 messages per second. In our local testbed, we found that the reported OWD is not affected by the number of clients, with median values between  $18.5\mu\text{s}$  and  $19\mu\text{s}$ , with standard deviations less than  $1\mu\text{s}$  and a maximum value of  $20\mu\text{s}$  across runs with a different number of clients. In contrast, figure 3 shows the OWD between one pair of VMs when varying the number of concurrent PTPd clients synchronising with the same PTPd master. The variations in the OWD when having up to four clients are not related to the number of clients. However, adding an additional client leads to an increase in the median latency of  $7\mu\text{s}$ . Having six or seven clients leads to increases in the OWD by approximately  $10\mu\text{s}$ . For eight clients, the median OWD is larger by approximately further  $7\mu\text{s}$ . For nine clients, the median OWD is approximately  $20\mu\text{s}$ . Additionally, in data centers from the two other providers we did not see any noticeable impact when using a maximum of four clients. We leave testing with more clients to future work. We hypothesize that this is related to the hypervisor, since

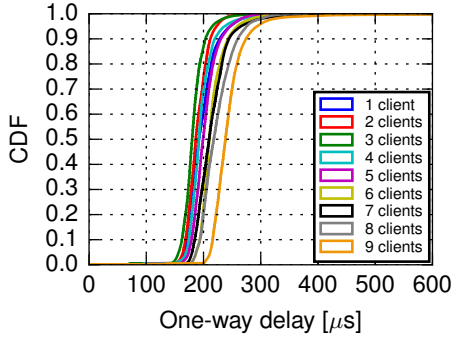


Fig. 3: Varying the number of PTPd clients synchronising with the PTPd master can affect the OWD.

our ten hosts testbed is composed of bare metal hosts, but the results are also influenced by the competing network traffic in the cloud, whereas in our testbed there was no other traffic. Given that our measurements were taken using a maximum of three simultaneous VMs, our measurements were not affected by this behaviour. To mitigate this issue, the current infrastructure of PTPmesh can be extended to perform measurements between VMs independently in pairs. Alternatively, all VMs can be visited in a round robin manner with the clock synchronization running for several minutes per client to allow the client’s clock to synchronise with the master. We leave this extension to future work.

#### IV. NETWORK LATENCY MEASUREMENTS

In the first instance, we set the number of Sync and Delay Request messages to 1 per second, since this is the default value configured in PTPd, which will be named in the rest of the paper as the low message frequency. We ran a full week of measurements in six data centers using the low message frequency. Additionally, we performed measurements in three data centers for one day using a higher message frequency of  $2^7$  messages per second, which will be named in the rest of the paper as the high message frequency. The challenge with using a higher message frequency is, on one hand, the increased CPU utilisation and network bandwidth at the end-host, while on the other, the amount of data collected for which additional storage is needed if measurements are performed for an extended period of time. The advantages of using a higher frequency are the detection of possible network congestion events with a higher resolution. Regardless of the message frequency used, the OWD values offered by PTP can serve as a reference for normal network conditions and can be used to detect anomalies.

A trace represents the PTPd statistics collected by the client, representing the measurements taken between two VMs. We list the number of messages recorded in the trace, and the start time and duration in Table II, each trace being identified by the assigned name of the data center and a number. For the first part of the table, the low frequency was used, while in the second part of the table, the high frequency was used. These

traces are indicative for the temporal perspective of network conditions in data centers. The spatial perspective is limited, since we used a maximum of three PTPd slaves at the same time. All datasets except one contain measurements taken between VMs that were located within the same zone. GCE-USW2-1 contains measurements taken between VMs that are located in different zones within the same region. Due to lack of space, we show figures only for representative traces.

##### A. Low Message Frequency

**Latency Values.** Table II lists the average, median, 99<sup>th</sup> and 99.9<sup>th</sup> percentiles, maximum, standard deviation for the OWD values, and the number of latency spikes (a sudden increase in latency to values over  $500\mu s$ ) for the trace. OWD values are higher in the EU data centers than the US data centers for EC2 and Azure. The GCE-EUW data center OWD values are similar to the ones in the GCE-USW data center, the difference coming from the extended period of increased latency that can be seen in Figure 4b. Most of the traces have maximum observed OWD values in the order of milliseconds.

In Figure 4a, in the EC2-EUW-2 trace we can observe multiple latency spikes, with a maximum observed of 14.364ms. In GCE-EUW the values are less or slightly higher than  $100\mu s$  up to the 90<sup>th</sup> percentile, with a maximum 99<sup>th</sup> percentile of 1.382ms and maximum 99.9<sup>th</sup> percentile of 3.477ms amongst the three VM pairs. In contrast, for GCE-USW data center, the maximum 99<sup>th</sup> is  $120.34\mu s$ , and only in the case of the trace between VM1 and VM2 the 99.9<sup>th</sup> percentile is higher,  $981.945\mu s$ , compared to the traces for the two other VM pairs. The traces captured in the GCE EU West data center stand out in comparison to the other traces collected, since they contain major disruptions for latency values over a prolonged period, accompanied by a high packet loss ratio. Between 2017-10-17 09:25 and 2017-10-18 05:16 the OWD reported varied greatly, reaching a maximum value of 8.83ms, with a significant part of the latency spikes of over  $500\mu s$  taking place during this interval. These events can be noticed for all three VM pairs, which can lead to the hypothesis that these events were data center-wide or that the VMs were placed within the same rack or on the same host. While the median latencies within the same data center are between 71 and  $97\mu s$ , the median latency between two data centers in the same region is  $180\mu s$  (GCE-USW2-1), almost double compared to the one ones within a data center.

The Azure UK West data center traces display a decrease of the OWD values of approximately  $100\mu s$  towards the end of the trace (Figure 4c), which corresponds to the network traffic for Sunday. The last part of the trace was captured on Monday, showing an increase in the OWD values back to the values before Sunday, except for the pair VM1-VM3. In the case of the Azure US West data center (Figure 4i) in the second day of measurements (after 172800 messages), a sudden decrease by approximately  $50\mu s$  in OWD can be noticed for all three pairs for a period of time, followed by an increase for the OWD to values higher by approximately  $50\mu s$  than the ones before the dip. It is interesting to see that

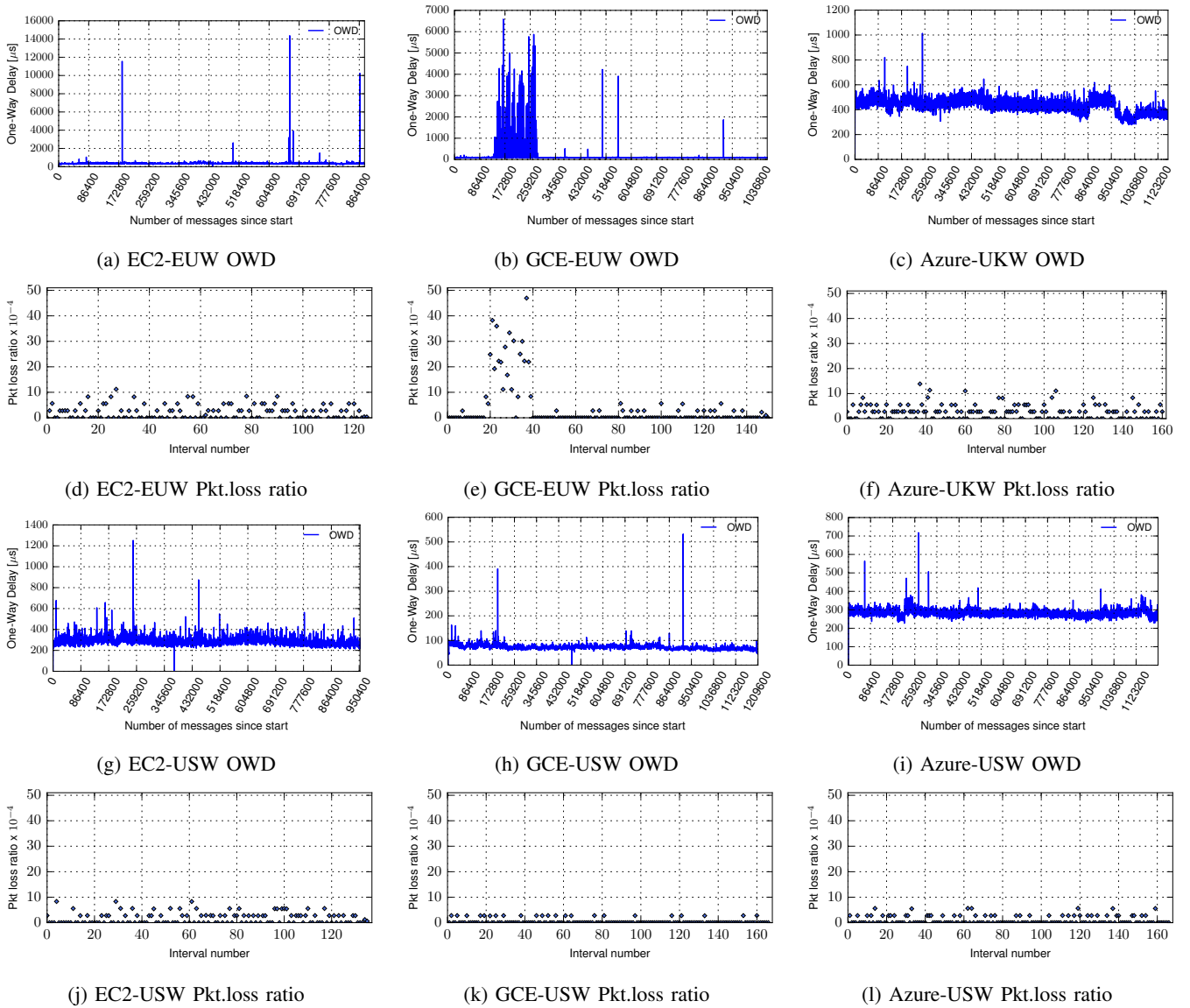


Fig. 4: OWD and packet loss ratios over 1-hour intervals between VM1-VM3 in EU and US data centers over one week.

the traces share similar characteristics for certain changes in the OWD values, meaning that the events were data center-wide or that the VMs were placed within the same rack or the same host. We performed experiments using better machines in Azure-UKS (Azure-UKS-N1), the median latency is in similar ranges to the ones obtained using slower machines. We additionally performed experiments with VMs with the new feature [2] enabled, which removes most of the software-based networking stack into FPGA-based smartNICs, and found that the one-way delay reported is significantly lower than the other reported values, with median values of  $94.54\mu s$  with low message frequency (Azure-UKS-A1) and  $82.28\mu s$  with high message frequency (Azure-UKS-A2). Recently, EC2 started offering a similar option using SR-IOV [19], but we have not performed measurements with it.

**Latency Variance.** An important aspect of network latency

is latency variance. If the variance is low, then the application performance will be determined by the median latency observed, essentially reducing the problem to improving the median latency in data centers. To this end, we compute the standard deviations of the one-way delay measurements over different intervals of time, and we call this the *latency variance profile* of the data center. The histograms in Figure 5 show the standard deviation of the OWD for intervals of 1 minute and 1 hour, binned in bins of size 1, truncated to 100. The distributions for the OWD are skewed to the right, towards small values, with a few values that are larger than the rest. The histograms for the two EC2 data centers are similar. When looking at periods of 1 minute, the standard deviations fall mostly between  $0\mu s$  and  $10\mu s$ . When looking at periods of 1 hour, the standard deviations fall between  $10\mu s$  and  $30\mu s$ . The histograms for the two GCE data centers are similar, but

TABLE II: Traces collected in data centers across the world from three cloud providers. Last column represents the number of latency spikes (l.s.) ( $> 500\mu s$ ) observed throughout the trace.

Trace	#Msgs	Start time	Duration	Avg.( $\mu s$ )	50 <sup>th</sup> ( $\mu s$ )	99 <sup>th</sup> ( $\mu s$ )	99.9 <sup>th</sup> ( $\mu s$ )	Max( $\mu s$ )	Std.dev.( $\mu s$ )	#L.s.
EC2-EUW-1	1204053	2017-11-06 14:43:20	7d00h03m10s	304.87 $\mu s$	289.57 $\mu s$	415.18 $\mu s$	516.5 $\mu s$	2.69ms	49.71 $\mu s$	19
EC2-EUW-2	879099	2017-11-06 14:43:22	5d15h12m50s	352.77 $\mu s$	345.17 $\mu s$	481.53 $\mu s$	2.32ms	14.36ms	192.57 $\mu s$	90
EC2-EUW-3	906750	2017-11-06 14:43:24	5d17h28m36s	352.17 $\mu s$	350.97 $\mu s$	459.5 $\mu s$	616.3 $\mu s$	3.16ms	50.68 $\mu s$	48
EC2-USW-1	934978	2017-11-07 12:56:48	5d10h08m30s	259.63 $\mu s$	256.93 $\mu s$	335.08 $\mu s$	486.3 $\mu s$	1.73ms	33.18 $\mu s$	7
EC2-USW-2	953109	2017-11-07 12:56:50	5d12h50m25s	279.22 $\mu s$	275.86 $\mu s$	361.42 $\mu s$	474.35 $\mu s$	1.25ms	29.08 $\mu s$	12
EC2-USW-3	870190	2017-11-07 12:56:52	5d01h04m16s	287.82 $\mu s$	283.44 $\mu s$	363.88 $\mu s$	429.06 $\mu s$	686 $\mu s$	24.15 $\mu s$	5
GCE-EUW-1	1208861	2017-10-16 14:11:06	7d00h00m00s	138.32 $\mu s$	97.1 $\mu s$	1.2ms	2.74ms	7.72ms	223.32 $\mu s$	237
GCE-EUW-2	1069898	2017-10-16 14:10:59	6d04h45m41s	138.29 $\mu s$	87.34 $\mu s$	1.44ms	3.48ms	6.58ms	268.65 $\mu s$	216
GCE-EUW-3	1208306	2017-10-16 14:10:51	7d00h03m09s	132.78 $\mu s$	82.97 $\mu s$	1.38ms	3.6ms	8.83ms	275.73 $\mu s$	243
GCE-USW-1	1210156	2017-10-16 16:38:32	7d00h02m57s	81.05 $\mu s$	76.9 $\mu s$	120.34 $\mu s$	981.94 $\mu s$	4.57ms	60.64 $\mu s$	16
GCE-USW-2	1210507	2017-10-16 16:39:15	7d00h02m14s	72.07 $\mu s$	71.35 $\mu s$	92.24 $\mu s$	119.22 $\mu s$	531 $\mu s$	8.65 $\mu s$	1
GCE-USW-3	1209171	2017-10-16 16:41:33	6d23h59m56s	79.7 $\mu s$	78.79 $\mu s$	104.34 $\mu s$	128.65 $\mu s$	396 $\mu s$	8.03 $\mu s$	1
GCE-USW2-1	42907	2017-04-07 23:58:42	0d05h59m28s	191.65 $\mu s$	180.66 $\mu s$	526.84 $\mu s$	908.27 $\mu s$	1.21ms	63.04 $\mu s$	5
Azure-UKW-1	1206919	2017-09-13 15:51:29	6d23h45m10s	441.37 $\mu s$	447 $\mu s$	529.27 $\mu s$	570.62 $\mu s$	1.38ms	47.4 $\mu s$	2380
Azure-UKW-2	1160593	2017-09-13 15:51:33	6d17h11m17s	432.95 $\mu s$	441.3 $\mu s$	522.88 $\mu s$	565.55 $\mu s$	1.01ms	48.7 $\mu s$	3979
Azure-UKW-3	1208739	2017-09-13 15:51:40	6d23h59m59s	412.59 $\mu s$	419.62 $\mu s$	483.48 $\mu s$	521.42 $\mu s$	827 $\mu s$	39.96 $\mu s$	134
Azure-USW-1	1203300	2017-09-13 15:26:09	6d23h08m41s	313.42 $\mu s$	315.14 $\mu s$	357.72 $\mu s$	379.86 $\mu s$	549 $\mu s$	22.78 $\mu s$	1
Azure-USW-2	1208955	2017-09-13 15:26:11	7d00h00m00s	282.46 $\mu s$	281.21 $\mu s$	330.64 $\mu s$	362.23 $\mu s$	717 $\mu s$	15.31 $\mu s$	3
Azure-USW-3	1208849	2017-09-13 15:26:16	6d23h59m59s	357.83 $\mu s$	358.46 $\mu s$	415.68 $\mu s$	449.11 $\mu s$	732 $\mu s$	22.22 $\mu s$	4
Azure-UKS-N1	108073	2018-02-22 20:11:00	0d16h37m07s	268.49 $\mu s$	261.31 $\mu s$	363.22 $\mu s$	481.65 $\mu s$	598 $\mu s$	25.16 $\mu s$	2
Azure-UKS-A1	96635	2018-02-22 22:18:24	0d13h54m09s	95.7 $\mu s$	94.54 $\mu s$	139.79 $\mu s$	212.75 $\mu s$	268 $\mu s$	11.08 $\mu s$	0
EC2-USE-1	21864606	2018-02-19 17:27:23	0d23h59m48s	181.96 $\mu s$	172.05 $\mu s$	291.55 $\mu s$	411.82 $\mu s$	2.04ms	30.71 $\mu s$	139
EC2-USE-2	21864606	2018-02-19 17:27:23	0d23h59m49s	197.33 $\mu s$	190.08 $\mu s$	293.74 $\mu s$	390.46 $\mu s$	1.77ms	27.14 $\mu s$	79
EC2-USE-3	21891242	2018-02-19 17:27:23	0d23h59m49s	188.53 $\mu s$	196.83 $\mu s$	301.86 $\mu s$	406.26 $\mu s$	1.48ms	30.65 $\mu s$	86
GCE-USW-1	19378393	2017-12-22 22:31:59	1d10h04m00s	65.48 $\mu s$	64.33 $\mu s$	89.08 $\mu s$	106.15 $\mu s$	451 $\mu s$	7.14 $\mu s$	0
GCE-USW-2	21161197	2017-12-22 22:32:17	1d09h17m34s	70.4 $\mu s$	69.47 $\mu s$	92.8 $\mu s$	106.11 $\mu s$	295 $\mu s$	8.35 $\mu s$	0
GCE-USW-3	20854143	2017-12-22 22:32:29	1d07h52m56s	58.47 $\mu s$	57.94 $\mu s$	76.02 $\mu s$	86.28 $\mu s$	286 $\mu s$	6.33 $\mu s$	0
Azure-UKS-1	20164042	2018-02-17 22:12:21	0d23h59m48s	286.58 $\mu s$	269.34 $\mu s$	684.45 $\mu s$	884.02 $\mu s$	1.22ms	74 $\mu s$	2235
Azure-UKS-2	21111652	2018-02-17 22:12:21	0d23h59m48s	271.41 $\mu s$	249.55 $\mu s$	724.43 $\mu s$	907.24 $\mu s$	1.37ms	84.65 $\mu s$	4747
Azure-UKS-3	17427943	2018-02-17 22:12:21	0d23h59m48s	340.02 $\mu s$	322.17 $\mu s$	760.13 $\mu s$	949.7 $\mu s$	1.29ms	81.7 $\mu s$	2793
Azure-UKS-A2	5043445	2018-02-23 12:47:02	0d05h54m20s	83.23 $\mu s$	82.28 $\mu s$	118.92 $\mu s$	178.79 $\mu s$	459 $\mu s$	9.11 $\mu s$	0

they are different from the two other cloud providers, with the standard deviations of the OWD values being smaller. For the US West data center trace, for 1 minute intervals, most of the values are between  $0\mu s$  and  $1\mu s$ . For 1 hour intervals, most of the values are between 1 and 10. The median values for the EU West data center trace are slightly higher, due to the increase of the OWD for a long period of time (1.5 days). There are differences between the two Azure data centers. In the case of the UK West data center, for 1 minute intervals, most of the values are between  $1\mu s$  and  $15\mu s$ . For 1 hour intervals, most of the values are between  $10\mu s$  and  $30\mu s$ . In the case of the US West data center, the values are slightly lower. For 1 minute intervals, most of the values are between  $0\mu s$  and  $10\mu s$ . For 1 hour intervals, most of the values are between  $10\mu s$  and  $20\mu s$ . It can be seen that in GCE the OWD has the lowest variance. EC2 and Azure are similar, with more variance seen for EC2. When enabling [2], the Azure latency variance profile becomes similar to the GCE one. Having less variance for OWD is better, since tail latencies can lead to a drop in application performance [20].

### B. High Message Frequency

**Latency Values.** The OWD values measured using high message frequency are lower than the ones measured using the low one. The EC2 US East OWD median values are between  $172\mu s$  and  $196\mu s$ . The GCE US West OWD values

have medians between  $58\mu s$  and  $69\mu s$ . Our low message frequency measurements (due to the low throughput, less than 20kbps) may have been redirected through switch gateways in GCE [18], whereas the high message frequency ones may have been sent host-to-host. After studying the timeline of the Azure-UKS OWD values, we noticed two prolonged periods of increased latency, with values close to and over 1ms. The medians OWD are between  $271\mu s$  and  $340\mu s$ .

**Latency Variance.** In the case of GCE-USW, the latency variance profile is similar to the one obtained using the low message frequency. Although the compared data centers are not the same, we note that the EC2-USE profile is similar to the EC2-USW one, and the Azure-UKS is similar to Azure-UKW. In the case of Azure-UKS, the tail of the profile is long, reflecting the two periods of increased latency that appear in the trace.

## V. PACKET LOSS MEASUREMENTS

We investigate packet losses in six data centers over a week for each of the VM pair. We logged the number of *Delay Request* and *Delay Response* messages exchanged between the slaves and the master when we ran the measurements with the low message frequency. Using the metric we defined for computing packet loss ratio in Section II, we compute the packet loss ratio over intervals of 1 hour and 1 day over one week, and we show the minimum, average, median, maximum



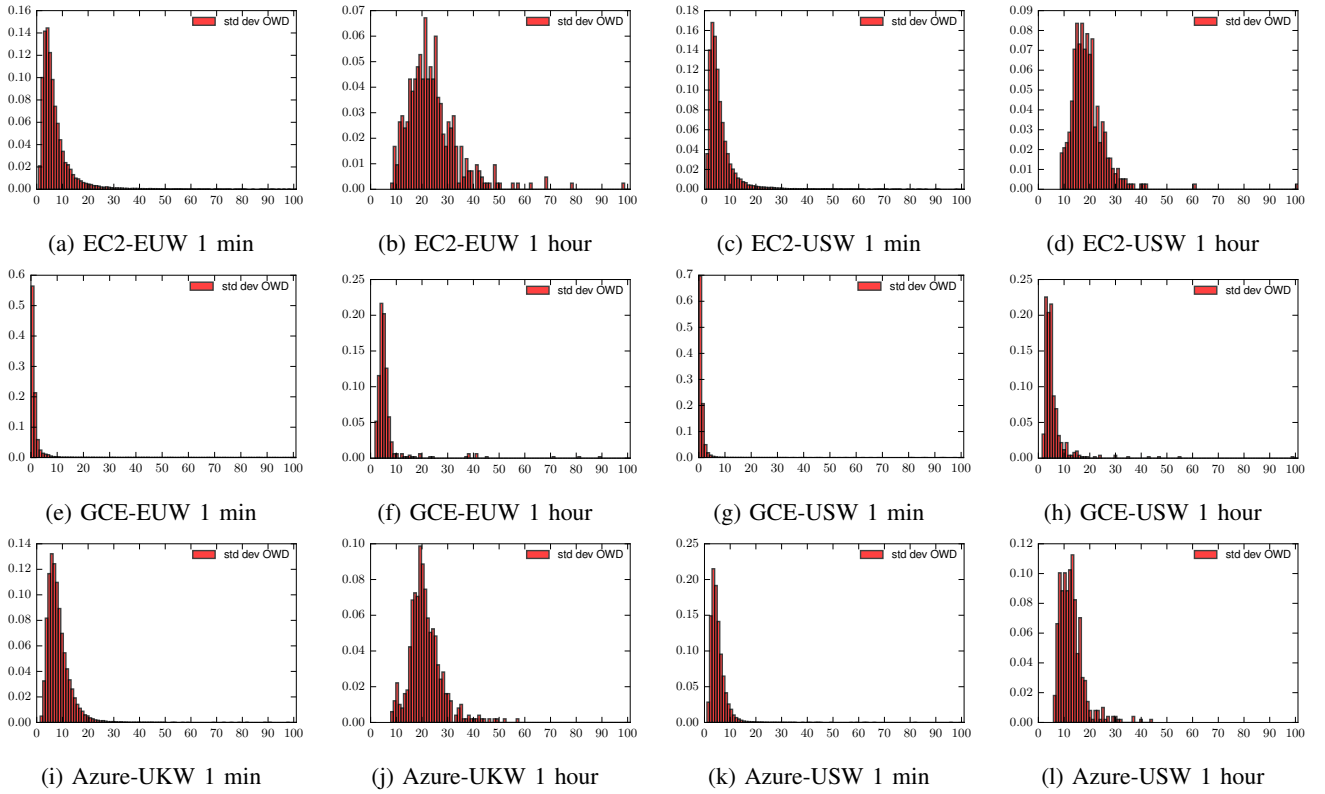


Fig. 5: Histogram of standard deviation values for OWD computed for different intervals in different data centers.

TABLE III: Packet loss ratio  $\times 10^{-4}$  over one week

Data center	1 hour					1 day				
	min	average	median	max	stddev	min	average	median	max	stddev
AWS-EUW	0.0	2.028	0.161	11.198	2.511	0.886	1.737	1.679	2.548	0.461
AWS-USW	0.0	1.06	0.0	8.324	1.74	0.116	0.779	0.777	1.617	0.373
GCE-EUW	0.0	2.96	0.0	46.961	7.081	0.109	2.95	0.463	14.082	4.56
GCE-USW	0.0	0.476	0.0	8.373	1.158	0.0	0.154	0.0	0.81	0.256
Azure-UKW	0.0	2.45	2.758	16.533	2.806	1.273	2.405	2.197	3.707	0.633
Azure-USW	0.0	1.244	0.0	11.123	1.9	0.116	0.843	0.753	1.618	0.417

and standard deviation for all the 1-hour and 1-day intervals in Table III. Interestingly, all EU data centers have higher packet loss ratios than the US data centers across all cloud providers. Figure 4 presents timelines over one week for packet loss ratios computed over 1 hour intervals for one VM pair in EU and US data centers, respectively. The packet loss ratio computed serves as a coarse-grained estimate and is influenced by the message frequency, but we can use it to obtain a baseline, and monitor anomalies.

In general, the packet loss ratios are low for all data centers, with most of the 1-hour intervals having no loss or having 1-4 messages lost per hour (out of 3600), which is at most approximately  $11.1 \times 10^{-4}$ . For EC2, the number of messages lost per hour is at most four (Figures 4d and 4j). High packet loss values of up to  $46.96 \times 10^{-4}$  appear in the first part of the GCE-EUW data center traces (Figure 4e), and are associated with the significant increase in network latency shown in Section IV-A, but later in the trace the values are normal, with

at most three messages lost per hour. In the GCE-USW data center (Figure 4k) the number of messages lost per hour is at most two, being the data center with the least packet loss. For Azure-UKW (Figure 4f), slightly higher packet loss ratios can be observed, while for Azure-USW (Figure 4l) the maximum number of messages lost per hour is four.

## VI. DISCUSSION

**Latency Contributors Analysis.** PTPmesh offers an end-to-end measurements, including the intermediate virtualization layer. The one-way delay latency values offer insights with respect to end-host overhead, in-network congestion and data center network architecture. If we couple our measurement results in data centers and our virtualization overhead measurements from Section III with the network latency contributions percentages presented in [4], we arrive at the same conclusion as prior research: the end-host, with the hypervisor, is a significant contributor to the overall measured latency from within

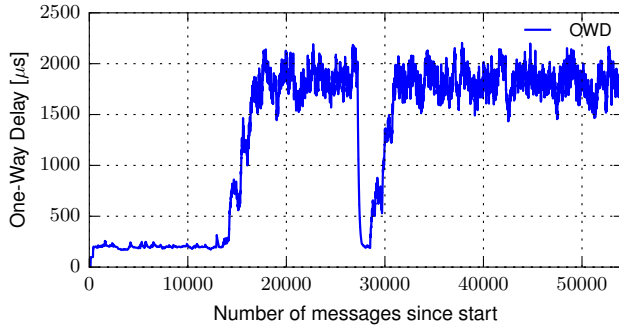


Fig. 6: Measured OWD between VM1 and VM10 in Amazon Ohio data center

the VM. The other significant contributor is network queueing. If we look at the network latency contributors [4], we notice that switching in the data center fat tree topology takes up almost 75%, which is approximately  $15\mu\text{s}$  in their analysis, while the rest is taken up by NICs and fiber length, with an estimate of  $20\mu\text{s}$  total. This analysis represents baseline contributions. On top of this, we can add the hypervisor’s overhead, which based on our measurements is  $200\mu\text{s}$  when using a low message frequency and  $190\mu\text{s}$  when using a high message frequency, giving a median baseline of  $220\mu\text{s}$  one-way delay, and  $210\mu\text{s}$  respectively. This back-of-the-envelope calculation shows that the remaining latency may come from in-network congestion, traffic bursts from other colocated VMs, transparent VM live migration, when it is observed for short periods, or from sustained increased network traffic (whose cause may be bulk network transfers across the data center, competing traffic from other colocated VMs, cluster drains), when it is observed over longer periods of time. Smaller values than this baseline mean that the OS is bypassed [2], [18], or that the VMs may be colocated on the same host, or that there is a shorter network path between VMs.

**Interesting Events.** While analyzing the datasets, we noticed a few interesting events. While performing the Amazon EC2 Ohio measurements, the latency suddenly increased substantially from median  $200\mu\text{s}$  to median  $1.75\text{ms}$ , as seen in Figure 6. We notice that the latency values return for 2 seconds (using the high message frequency) to previous values, but then the latency increases again.

In the Azure-KS data center, after restarting our VMs, we consistently got substantial latencies (median  $1.39\text{ms}$ ) compared to previous values (median  $191.49\mu\text{s}$ ), that we kept on measuring even after several VM restarts, and across almost one month of measurements. The first time we observed these large latency values was on the 29th of December 2017, and the last time we performed measurements in this data center was 23rd of January 2018.

## VII. CONCLUSION

Data centers have become an essential computing infrastructure, whose network complexity needs specialized monitoring

tools to ensure their performance and reliability. In this paper, we extended our work on measuring network latency and packet losses in data centers using PTPmesh. We offered a better understanding of how to use PTPd data, and we performed a detailed analysis of the measurement data collected in several data centers, finding that data centers have different latency, latency variance and packet loss characteristics within the same provider and across providers. We provide our data to the community. In future work, we would like to explore extending our current infrastructure for fault localization.

**Acknowledgements.** The authors would like to thank George Neville-Neil, Salvador Galea and Noa Zilberman for helpful discussions and logistic support. This work is supported by the EU FP7 ITN METRICS (grant agreement no. 607728) and the EPSRC under EARL (project reference EP/P025374/1).

## REFERENCES

- [1] A. Singh *et al.*, “Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network,” in *Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2015.
- [2] D. Firestone *et al.*, “Azure Accelerated Networking: SmartNICs in the Public Cloud,” in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018.
- [3] J. C. Mogul and R. R. Kompella, “Inferring the Network Latency Requirements of Cloud Tenants,” in *Hot Topics in Operating Systems (HOTOS)*. USENIX, 2015.
- [4] N. Zilberman, M. Grosvenor, D. A. Popescu, N. Manihatty-Bojan, G. Antichi, M. Wojcik, and A. W. Moore, “Where Has My Time Gone?” in *Passive and Active Measurement (PAM)*. Springer, 2017.
- [5] D. A. Popescu *et al.*, “Characterizing the impact of network latency on cloud-based applications’ performance,” University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-914, Nov. 2017.
- [6] C. Guo *et al.*, “Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis,” in *ACM SIGCOMM*. ACM, 2015.
- [7] Y. Zhu *et al.*, “Packet-Level Telemetry in Large Datacenter Networks,” in *ACM SIGCOMM*. ACM, 2015.
- [8] Y. Peng, J. Yang, C. Wu, C. Guo, C. Hu, and Z. Li, “deTector: a Topology-aware Monitoring System for Data Center Networks,” in *Annual Technical Conference (ATC)*. USENIX Association, 2017.
- [9] D. A. Popescu *et al.*, “PTPmesh: Data Center Network Latency Measurements Using PTP,” in *Symp. on Modeling, Analysis, and Simulation of Computer and Telecom. Systems (MASCOTS)*. IEEE, 2017.
- [10] “PTPd,” <https://github.com/ptpd/ptpd>, online; accessed May 2018.
- [11] “IEEE 1588-2008 Precision Time Protocol,” <https://www.nist.gov/el/intelligent-systems-division-73500/introduction-ieee-1588>, online; accessed March 2017.
- [12] “Solarflare PTP Adapters,” <http://www.solarflare.com/ptp-adapters>, online; accessed March 2017.
- [13] B. Arzani *et al.*, “Taking the Blame Game out of Data Centers Operations with NetPoirot,” in *Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2016.
- [14] A. Roy, H. Zeng, J. Bagga, and A. C. Snoeren, “Passive Realtime Datacenter Fault Detection and Localization,” in *Networked Systems Design and Implementation (NSDI)*. USENIX, 2017.
- [15] B. Arzani *et al.*, “Closing the Network Diagnostics Gap with Vigil,” in *Special Interest Group on Data Communication (SIGCOMM)*. ACM, 2017.
- [16] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, “High-resolution Measurement of Data Center Microbursts,” in *Internet Measurement Conference (IMC)*. ACM, 2017.
- [17] R. Durairajan, S. K. Mani, J. Sommers, and P. Barford, “Time’s Forgotten: Using NTP to Understand Internet Latency,” in *Hot Topics in Networks (HotNets)*. ACM, 2015.
- [18] M. Dalton *et al.*, “Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization,” in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018.
- [19] “Amazon Enhanced Networking,” <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html>, online; accessed May 2018.
- [20] J. Dean and L. A. Barroso, “The tail at scale,” *Commun. ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013.